Service Selection using Non-Functional Properties in MANETs

K. Ponmozhi[1] and Dr. R.S. Rajesh[2]

[1]Lecturer,Department of Information Technology,Hajee Karutha Rowther Howdia College, India.

Email: arulchezhiyan96@gmail.com

Department of Computer Science & Engineering, Manonmaniam Sundaranar University, Tirunelveli, India

Email: rsrajesh_tvl@yahoo.co.in

***Abstract*- Mobile ad hoc networks are the ones which allow mobile nodes to spontaneously form a network and share their services. The dynamic environment of MANETs demands service selection should not only based on functional properties but also be driven by non-functional requirements. In this paper we present a modelling method of Non-Functional properties. The degree of impact of the property on QoS may vary so, we allow the application designer to define weight for each property. The evaluation function for the properties cannot be uniform as the type of the property may be Boolean, string or numeric depending upon the nature of the property. We define three different.**

***Index Terms*—Mobile Ad hoc Networks, Non-functional properties, Service Provisioning, Service selection.**

I. Introduction

Ad hoc networks are distributed networks of mobile nodes without any fixed infrastructure. In these networks the nodes involved have to provide and access services of each other. Service means either hardware of software which provides services to others in the network. In the case of MANETs most of the devices involved are resources constraint, which leads the need of finding only the relevant services. So far there is no automatic selection of services among different providers of the same service.

Service Oriented Architecture has been emerged as a solution for distributed systems, and they are suitable especially for loosely coupled system. SOAs enable modularizing the more complex systems in a way that they are composed of independent software components that offer services to one another through well defined interfaces.

The advantage of SOA architecture is invocation using late binding (i.e.) binding taking place at the time of execution. One among the major step in SOA implementation is finding a service. Match making is done only on the functional properties of the service. In the case of mobile users they need services that are relevant to their current situation. They prefer services which are nearer to them with up-to-date information [2] like news, railway enquiry reply. This mandates the consideration of the non-functional properties such as context properties which specify the current situation of the user will be the suitable one to achieve better performance and user satisfaction.

Service selection becomes a complex task if we need to consider many functional and non-functional properties. The issues of concern in service selection are:

- How to specify service requirements
- How to evaluate the services provided based on the specified requirements and brings out single aggregated value.

Non Functional requirements of a service denote all the aspects which can be used by clients in order to evaluate service quality; they play an important role to differentiate among services of same functionality but which differ with respect to the user's current situation [6]. In this paper we propose methods to formalize and send Non Functional properties along with the service functional descriptions. We discussed methods of analyzing and matching the attributes to select services. The paper is organized as follows section 2 briefs the related work, section 3 gives the service architecture. section 4 classifies the Non-functional properties section 5 details the modeling of NFPs section 6 specifies the evaluation and selection finally in section 7 conclusion and future work which completes the paper.

II. Related work

Non-functional properties are used as filtering mechanism to find a best match among the choices of services. They increases the rich information provided as a form of pre-requisites for automated service discovery and selection. In [1], authors' facilities web service discovery process which fulfils their needs using enhancement in UDDI. The work in [3] is a middleware which supports multi-dimensional QoS. It focuses on optimizing over all QoS provided by a omposition of service endpoints. The key difference between the other work is we assign values based on the data type supported and each type of attribute is evaluated differently thus gives a general evaluation scheme.

III. Service architecture

In order to specify the services we can categories the properties as functional and non-functional properties based on the technical information and others. Functional properties are input, output, operations provided, how to access these services etc Non-functional properties are the one which specifies the quality of the services which are similar to "adjectives". These can be used to define the quality of the service as well as goodness of the services for example the price, performance; bandwidth the consumption etc. of the service. The QoS service incorporates requirements of the

DOI: 03.LSCS.2013.1.577

consumer, provider, and the network participants [4]. On the one hand the provider can use these properties to specify the services' quality. On the other hand the client/requester uses them to specify their constraints.

Service providers describe their services and advertise them. These service descriptions will be received by the nodes. They will store them in the repository. The request will be formed by the requester by using the same schema which is used to define by the provider. The application designer will assign values for the attributes based on the need of the application. The requesters current context can be accessed transparently form the user and device profile [5]. The values on the request will be matched against the services' values and ranking will be done in order to select most relevant service.

IV. Classification of Non-Functional Properties

Any service can be described by its functional and non-functional properties. Functional properties are those properties which define the technical aspects of the operations it provide. Non-functional properties are properties which define all aspects which can be used by client in order to specify the service quality.

We classify the Non-Functional Properties as: Quality of Service, and Context properties.

The *QoS properties* are used to specify the service quality that will be provided by the particular service like cost, performance, reliability etc. Each application will have its own set of priorities on the properties. For example Mission-critical applications may prioritize energy efficiency and speedy service response time, where the applications like building automation may prioritize monitoring quality and network utilization. A same set of services' constraints cannot satisfy every application. So we should provide mechanism to specify the weighing factor for each property depending upon the need.

Context Properties are properties that reflect the current context of the application, client and the service provider. Context is information that can be used to characterise the situation of an entity. They can be further grouped into (static) domain specific and dynamic. The domain specific properties are the one which may vary for each application domain. In a particular application we may need color printing, at some application we may need laser printing instead of dot-matrix printing. These can be captured in the design time itself, and thus can be defined as static context properties and can be captured at the time of application design time and can be described at the time of service description itself. Dynamic properties are the one for which the values can be captured only at runtime such as battery power; load of the server, moving sped of the node etc.

V. Modelling Non functional properties

Modelling the non functional properties means representing the NFPs of services using any language or using some structured way. The model defined will be used by both the service providers to describe their service qualities and consumers to specify their constraints in a convenient way. The desirable criteria of the model are: (i) It should support for new addition of properties. (ii) Since the data type of each property may vary the evaluation function should take into consideration of this and the sorting based on this model must be generic even if we add new properties. (iii) The model should facilitate the user to specify their preferences for each of the NFPs depending upon the situation. (iv) The functional and non-functional description of services should be systematically separated. (v) The description of the service should not be restricted to a single application domain.

We define schema for the service representation and assume that the same schema will be used by the client to specify their requirements. The requirement for a service may vary from application to application. For example if the application needs a financial service then the security is having highest importance where as if the application needs a printing service for which low cost is preferable than security. Similarly it may depends on the user also, i.e one user may want good quality but she/he may not bother about cost, some user may need less cost as highest preferable one than the quality. So we need to allow the application to specify its own requirements. In order to facilitate this, we assign every operation of a service to a category, for example printing service. Each category will have a set of properties assigned to it. Each property is defined as a set of four values {<name>, < type>, <weight>, <value>}.

Every property will be of different data types. For example the property color in the case of printer may have yes/no i.e Boolean value, where the mechanism should be one among the following laser, dot-matrix. So we provide facilitation to specify the type of the data this property will have.

Each property will have its own impact factor if available to an application. For example at some time quality will be more important than cost, at some time the otherwise. So, we provide facility to assign weight to each of the property by following weighing table. Weighing table we use gives the user the freedom of specifying one among ten preferences provided it satisfies $\sum|wi| = 1$; where i = 1 to n for n properties. Based on the weighting value we differentiate the properties as *Mandatory* and S*econdary* properties. The mandatory properties will have weight as 1, which is the maximum value for weight. It means the satisfaction of this property is essential, failing which the service cannot be utilized. So this is used to filter the services. The value can be either positive or negative. Positive value represents the higher value in this attribute is preferable, and negative represents the lower value is preferable. For example the property performance should be high where as we need less cost.

The Value attribute may take any value based on the data type of the attribute. It is used to specify the value expected to be for this particular attribute. For example if the data type of the attribute is Boolean, then it can have either yes or no to be matched, e.g the property color will have yes/no as its value. The value of the Boolean and string type are used to

DOI: 03.LSCS.2013.1. 577

match with the value specified in the service's properties exactly. But the value specified in numeric type is used somewhat differently. Consider this situation where we need less cost for the printing service. If we specify value 10 to the value it does not mean that we don't want printing service lesser than that price. If lesser than that is available then we prefer that, but the cost should not exceed what we specified. So, for numeric evaluation we find the services which provides lesser value if the weight is negative. So, we compare all the selected services and find the position in which every service's value belongs to among the availability. The value property of the numeric data type is used as filter, i.e if the weight is -.6 or any numeric attribute it means we assign the preference as .6 but the value we expect is lower (as the weight is negative), and if the value attribute is specified or example value is 20, then we filter all the service which are higher than this, because we need services which provides lesser or equal to the value specified. While calculating the metric we omit this value. As we defined the schema for the attribute value specification we can allow the user to select the values one among the set of values. This can be done if specify the data type as enumeration. If the number of matching element is high then the service's metric is high.

VI. Assessing the services

The importance of the attributes for a service may vary for different clients. For example one client may prefer to have speedy reply where as other may be keen on the location of the service. We provide the freedom of specifying importance of an attribute by the application designer itself. Since we allow different data type for the attributes. The calculation varies slightly for each attribute.

For Numeric type : The data types integer, double and currency comes under this. In our example price is of type currency. In the case of data like bandwidth, price etc. though they are numeric data we need the lower value. If the weight of any attribute is less than -1 then it means that the lower value on this is expected.

We calculate the metric of the provider by comparing with the maximum and minimum possible values that can be provided for this attribute in the market. If "Smax" is the maximum value and "Smin" is the minimum value for this attribute the we calculate the metric of a service provider as

For a positive weight

metric = 1 - (Smax – value) / (Smax – Smin);

For the negative weight

Metric = (Smax – value) / (Smax – Smin)

For Boolean type of attribute we will do the exact match on the value with the service's value.

If there is a match then metric = 1, else 0.

For the enumeration type data the string specified in the "value" will be matched with the service's value.

Metric = (v1+v2+..+vn)/n vi = 1 if a match else 0

For a match we add 1 to and for mismatch we add 0 and the final score will be divided by the number of elements in the set. Here in our example the payment attribute of the service will be matched against the value of the payment in the request. If the service provides "credit card" alone then the value will be (1/2) which is .5, if it matches both credit card and debit card then (1 + 1)/2 i.e. 1 is the credit for this attribute.

Suppose say service "a" provides { color printing, credit card and debit card payment and the price is 5 } We calculate the metric as follows:

For the first attribute "color" which is a Boolean attribute, as it matches exactly metric for the first attribute is 1.

For the second attribute "payment" the enum attribute, we find the service provides both the requirement of the requester so we have 1 as explained above.

For the third attribute "price" which is a numeric attribute, and the weight specifies that we need lower value for this attribute that is we need a service which prints for lesser price. If the maximum printing price is 20 per page and the minimum is 2 per page then the price we provide satisfies the customer to .833 ((20-5)/(20-2)).

Color metric(m) = 1 ; payment metric(m) = 1, and price(m) is .83 Like this we will calculate the metric for each attribute specified in the request against the service's values. To find the metric of the service by the provider the metric values will be multiplied by the weight specified assuming that the sum of weight will be equal to 1.

VII. Service selection

During service discovery phase, the services will be compared with the queries. After populating services based on the functional requirements of the user, if there are more than one services for the given request, for the inclusion in the selection list each service should satisfy all the mandatory constraints. If any one of them is not satisfied then the service will be filtered out. Then we have to select the service based on their preferable constraints.

Mandatory attributes are used to filter out the services if there is no exact match. The weight for these attributes will be assigned as 1. In the case of numeric type the weight can have 1 and the value attribute to some non-negative value to represent that the attribute must have value less than the value specified in the value attribute of the requirement. For example if the requirement is "the cost must be less than or equal to 50"

Step 1*: Filtering* the services based on mandatory attributes.

Case 1: Numeric type property with weight is negative and value specified

Select the service if its value it provides is less than or equal to the value specified in the request.

Eg. { Name="cost" weight= -.6 value=50 type = numeric}. The services which provides less or equal to 50 only will be added to the list for further processing.

Case 2: Boolean type with weight 1 (value must be present)

Select the service if its value is exactly equal to the one specified in the request.

Eg. {Name="security" weight=1 value="high"

DOI: 03.LSCS.2013.1.577

type=Boolean}. The services which are providing high security will be added to the list for further processing.

Case 3: Enumeration type weight =1 (value must be specified) Select the service if all the items in the requirement set matches to the availability, else 0.

Step 2: *Evaluate* on the weights assigned and order the services. Sum (Mi*Wi) will be calculated for all the secondary properties, and sorted based on the overall value.

VIII. Experiment Results

To understand the feasibility of the aspects, we have implemented the functions. We used 10 parameters with the preference specified. We assumed that 5 same service instances exist and the provider's values also specified. The request is assigned with some preference value and the result shows that the specification of weight affects the selection of the services. Table I list the 5 services with the property values they provide.

Table I. Sample services

#	Color	Price	Available	Location	Performance	Reliability	Security	resolution	No. in queue	File type
S1	yes	3%	yes	X	.8	.6	High	high	No	Pdf,doc,bmp
S2	No	2%	yes	A,Y	.5	.5	high	high	No	Pdf, doc, bmp
S3	yes	5%	No	Z	.9	.8	medium	medium	Yes	doc, bmp
S4	no	2%	no	X,Y,A	.9	.9	high	high	No	Pdf, doc, bmp
S5	yes	4%	yes	B,A	.5	.6	High	low	Yes	doc, bmp

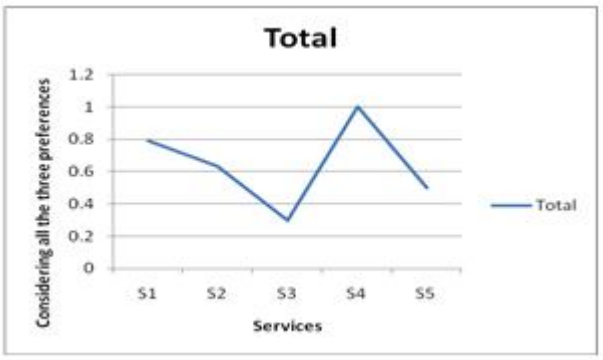

Fig. 1. Considering all

Table II. Represent the total value considering the following request

{Name="Queue" weight=.5 value=NO type=Boolean}
{Name="Location" weight=.2 value="'X,Y,Z" type=Enum}
{Name="Performance" weight=.3 type=Numeric}

From table II we can find when we consider only performance S3 and S4 are having equal values, where as considering location alone S4 is at the top and S2 comes next. When we consider all the requirements with varying preferences weights the value will be different. The calculation is given in table II. From Fig. 1 we can say that S4 is the service on the top then, S1, S2, S5 and finally S4. So, if we specify the request with the preferences for all the required properties we can select the appropriate service for our need.

Conclusion and Future work

Due to the increased popularity of web service technology and the availability of may providers for a same service increase. The consumers are therefore concerned about finding services that are relevant to the current context. This paper proposed and exemplifies the influence of the non functional properties in service selection. The model proposed can be used for capturing all the non functional properties, adding new one if need arises without making any major work.

Table II. Total metric values

	Performance			Location			Queue			Total
#	Weight .3	EF .75	W*EF .225	Weight .2	EF .33	W*EF .066	Weight .5	EF 1	W*EF .5	.791
S1										
S2	.3	0	0	.2	.67	.134	.5	1	.5	.634
S3	.3	1	.3	.2	0	0	.5	0	0	.3
S4	.3	1	.3	.2	1	.2	.5	1	.5	1.0
S5	.3	0	0	.2	.33	.066	.5	0	0	.5

References

[1] Amal Yousief, Hesham Arafa, and Ahmed Saleh, A future Image for Web Services' Discovery with a Client Web based Inerface in International Journal of Computer Applications, vol35, No.10, Dec 2011.

[2] Doulkeridis C. And Vazirgiannis M. , A System Architecture or context-aware service discovery, 2005.

[3] N.B. Mabrouk, S.Beauche, E.Kuznetsova, N.Georgantas, and V. Issarny. QoS-aware service composition in dynamic service oriented environments. In Proceedings of Middleware, Middleware'09, pp7:1-7:20, 2009.

[4] Chien-Liang Fok, Christine julien, Gruia-Catalin Roman, and Chenyang Lu, Challenges of satisfying multiple stakeholders: Quality of Service in the internet of Things, SESENA'11 , May 22,2011, Waikiki, Honolulu,HI,USA.

[5] K.Ponmozhi and R.S. Rajesh, Bringing context awareness into MANETs" at the National on "Explorations & iNnovations in Advanced Computing, National Engineering College, Kovilpatti, India,2008,pp.138-144

[6] Kristof Hmann, Sebastian Steenbuck, and Sonja Zaplata, Towards NFC-aware Process Execution for Dynamic Environments, Proc. WowKiVS 2011,EASST Vol 37(2011)

DOI: 03.LSCS.2013.1.577